REMARKS/ARGUMENTS

Claims 1-20 are pending in the present application. Claims 1, 7, and 13-17 are amended. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 101

The Final Office Action has rejected claims 13-16 under 35 U.S.C. § 101 as being directed towards non-statutory subject matter.

Regarding this rejection, the Final Office Action states:

> Claims 13-16 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. The claims are directed to a signal directly or indirectly by claiming a medium and the Specification found in paragraph 88 recites evidence where the computer readable medium is define as a "radio frequency and light wave". In that event, the claims are directed to a form of energy which at present the office feels does not fall into a category of invention. The following link on the World Wide Web is for the United States Patent And Trademark Office (USPTO) policy on 35 U.S.C. §1 01.

Final Office Action dated May 29, 2008 p. 2.

The Applicant has amended claim 13 to recite a "computer program product in a computer readable recordable-type medium..." Support for the amendment can be found on page 36 of the specification. In light of the amendments to the claim, the Applicant submits that the rejection is now moot. Withdrawal of the rejection is therefore respectfully requested.

II. 35 U.S.C. § 102, Anticipation

The Final Office Action has rejected claims 1-3, 5-6, 13-16 under 35 U.S.C. § 102 as being anticipated by *Alfieri,* U.S. Patent No. 5,745,778 (hereinafter "*Alfieri*"). This rejection is respectfully traversed.

Regarding this rejection, the Final Office Action states:

> As per claim 1, Alfieri teaches a method of queuing threads among a plurality of processors in a multiple processor system having a plurality of multi-processor modules, wherein each of the plurality of multi-processor modules comprises a plurality of processors (Fig 1: units 100-103 form one processor module and units 104-107 form another processor module), wherein each of the plurality of multi-processor modules is associated with one of a plurality of chip run queues, and wherein each of the plurality of processors is associated with one of a plurality of local run queues (Column 9, lines 40-52: level 1 queue would correspond to the chip run queue and level 0 queues for each processor corresponds to local run queues), the method comprising the computer

> implemented steps of: receiving a first thread to be processed; identifying the first thread as part of an existing process on a first multi-processor module of the plurality of multi-processor modules (Column 6, lines 31-36; Column 9, lines 55-57; Column 10, lines 1-3); performing a search for an idle processor, wherein the search is restricted to the plurality of processors of the first multi-processor module associated with the existing process (Column 6, lines 31-36; Column 9, lines 58-61) assigning the first thread to either one of the plurality of chip run queues, or one of plurality of local run queues (Column 9, lines 62-64).

Final Office Action dated May 29, 2008 p. 3.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). In this case, each and every feature of the presently claimed invention is not identically shown in the cited reference, arranged as they are in the claims.

Alfieri teaches a method of assigning threads among the processors of a multi-processor system. According to *Alfieri*:

> The system uses a hierarchy of processing levels and run queues to facilitate affining thread groups with processors or groups of processors when possible. The system will tend to balance out the workload among system process and will migrate threads up and down through processing levels to increase cache hits and overall performance. The system is periodically reset to avoid long term unbalanced operation conditions.

Alfieri, Abstract.

Alfieri accomplishes this by setting out various priority processing rules for the assignment of threads. In the system of *Alfieri*, when a processor becomes idle, it examines its own level 0 queue (i.e., a local run queue), the associated level 1 queue (i.e., a chip run queue), and an associated level 2 queue (i.e., a global queue). *Alfieri*'s idle processor thus identifies a thread having a highest priority from among all 3 of these queues in a greedy algorithm type manner.

When the highest priority thread is identified, *Alfieri* then attempts to relocate the thread into a lower level cache in order to increase the "cache locality." *Alfieri* explains this in the following passage:

> When CPU 100 becomes available to execute a thread (step 501), it looks (step 502) at its own Level 0 run queue, the Level 1 run queue for its associated cache 110 and the Level 2 run queue. If one or more thread groups are available, CPU 100 proceeds to check for the highest priority thread group (step 519). If the highest priority level is shared by thread groups on different run queues, CPU

> 100 will break the tie (step 521) by selecting the run queue to use in a round-robin fashion.
>
> Once a thread group is selected, CPU 100 will begin the process of determining if the thread group can be moved closer to the CPU.

Alfieri, col. 7, ll. 14-25.

While *Alfieri* does attempt to move the threads to a more local queue in order to improve the cache locality of the threads, it is easy to see that the method of *Alfieri* will process the most immediate priority thread without regard to process switches and context switches. That is, because an idle processor of *Alfieri* looks only for the highest priority thread, the individual processors of *Alfieri* will continuously switch processes and contexts in order to process the higher priority threads regardless of the present affinity of the chip or the processor.

In contrast, claim 1 as amended, seeks to maintain chip and processor affinity by preferably assigning threads of a common process to the same local queue for execution on the same processor. That is, when Applicant's new thread is assigned to a queue, Applicant's first determine whether the thread is part of an existing process, and then will assign the thread directly to the lower level caches. The Applicant's process therefore tries to minimize the process switches and context switches that are of no concern to *Alfieri.*

Claim 1 has been amended. Support for the claim 1 amendments can be found on pages 20-22 of the specification, and in Figure 5. Claim 1, as amended, is as follows:

> 1. A method of queuing threads among processors in a multiple processor system having a plurality of multi-processor modules, wherein each of the plurality of multi-processor modules comprises a plurality of processors, wherein each of the plurality of multi-processor modules is associated with one of a plurality of chip run queues, and wherein each of the plurality of processors is associated with one of a plurality of local run queues, the method comprising the computer implemented steps of:
>
> receiving a first thread to be processed, wherein the first thread belongs to a first process;
>
> identifying whether the first thread is a bound thread associated with a first processor of the plurality of processors;
>
> responsive to identifying that the first thread is the bound thread associated with the first processor, assigning the first thread to a first local run queue of the plurality of local run queues, wherein the first local run queue is associated with the first processor;
>
> responsive to not identifying that the first thread is the bound thread, identifying whether the first thread is part of an existing process on a first multi-processor module of the plurality of multi-processor modules;
>
> responsive to identifying that the first thread is part of the existing process on the first multi-processor module, attempting to identify an idle processor that has executed a second thread belonging to the first process, wherein the idle processor is identified from the plurality of processors of the first multi-processor module associated with the existing process;
>
> responsive to identifying that the idle processor that has executed the second thread belongs to the first process, assigning the first thread to a second local run queue

of the plurality of local run queues, wherein the second local run queue is associated with the idle processor; and

responsive to not identifying that the idle processor that has executed the second thread belonging to the first process, assigning the first thread to a first chip run queue, wherein the first chip run queue is associated with the first multi-processor module.

II.A. Claim 1

II.A.1. *Alfieri* does not teach "identifying whether the first thread is a bound thread associated with a first processor of the plurality of processors" and "assigning the first thread to a first local run queue of the plurality of local run queues"

Alfieri does not identify whether a thread is a bound thread that should be executed at a specific processor. As pointed out above, *Alfieri* only determines the current highest priority thread, and then attempts to execute that thread in a greedy fashion. Idle processors pull high priority threads into their local queues regardless of the current context and process of the thread and local processor. Thus, threads are executed without consideration of the current context of the idle processor.

Alfieri does not mention bound threads, nor does the greedy thread allocation of *Alfieri* offer any way to process a bound thread. Therefore, *Alfieri* does not disclose either of the claim 1 features of "identifying whether the first thread is a bound thread associated with a first processor of the plurality of processors," and "responsive to identifying that the first thread is a bound thread associated with a first processor, assigning the first thread to a first local run queue of the plurality of local run queues, wherein the a first local run queue is associated with the first processor." Because *Alfieri* does not teach these claim features, *Alfieri* cannot properly anticipate amended claim 1 under 35 U.S.C. § 102.

II.A.2. *Alfieri* does not teach "identifying an idle processor that has executed a second thread belonging to a first process" and "assigning the first thread to a second local run queue of a plurality of local run queues, wherein the second local run queue is associated with the idle processor"

Similar to the claim features shown in II.A. above, the cited claim features here minimize the context and process changes by assigning a thread to those processors and queues that have recently processed other threads of the same process. The greedy thread allocation of *Alfieri* instead assigns threads to the global queue, and then pulls those threads down into chip and local queues, without consideration of the current context of the idle processor.

Alfieri does not disclose either of the claim 1 features of "attempting to identify an idle processor that has executed a second thread belonging to a first process, wherein the idle processor is identified from the plurality of processors of the first multi-processor module associated with the existing process,"

and "responsive to identifying the idle processor that has executed a second thread belongs to a first process, assigning the first thread to a second local run queue of a plurality of local run queues, wherein the second local run queue is associated with the idle processor." Because *Alfieri* does not teach these claim features, *Alfieri* cannot properly anticipate amended claim 1 under 35 U.S.C. § 102. Withdrawal of the rejection is therefore respectfully requested.

II.B. Claims 2, 3, 5-6

Claims 2, 3, 5, and 6 depend from claim 1. The rejections of claims 2, 3, 5, and 6 are predicated upon the assertions made with respect to independent claim 1. As shown above, the underlying assertions made by the Examiner regarding *Alfieri* are incorrect vis-à-vis independent claim 1 as amended. Therefore, *Alfieri* does not teach all of the features of claims 2, 3, 5, and 6 at least by virtue of their dependence on independent claim 1. Therefore, the rejection of claims 2, 3, 5, and 6 under 35 U.S.C. § 102 has been overcome. Withdrawal of the rejections is therefore respectfully requested.

II.C. Claims 13-16

Claim 13 has been amended to recite features similar to those found in claim 1. The remarks presented above with regard to claim 1 are equally applicable to claim 13. By virtue of the similarities found therein, and the remarks presented above, *Alfieri* does not anticipate applicant's claims 13 as amended. Therefore, the rejection of claims 13 under 35 U.S.C. § 102 has been overcome.

The rejections of claims 14-16 are predicated upon the assertions made with respect to independent claim 13. As shown above, the underlying assertions made by the Examiner regarding *Alfieri* are incorrect vis-à-vis independent claim 13 as amended. Therefore, *Alfieri* does not teach all of the features of claims 14-16 at least by virtue of their dependence on independent claim 13. Therefore, the rejection of claims 14-16 under 35 U.S.C. § 102 has been overcome. Withdrawal of the rejections is therefore respectfully requested.

III. <u>35 U.S.C. § 102, Anticipation</u>

The Final Office Action has rejected claims 7-12, and 17-20 under 35 U.S.C. § 102 as being anticipated by *Kimmel et al.*, U.S. Patent No. 6,105,053 (hereinafter "*Kimmel*"). This rejection is respectfully traversed.

Regarding this rejection, the Final Office Action states:

> As per claim 7, Kimmel teaches a method of load balancing threads among processors in a multiple processor system having a plurality of multi-processor modules (Fig 1A, units 10, 11, 12; Fig 1B, units 110, 111), wherein each of the plurality of multiprocessor modules is associated with one of a

> plurality of chip run queues, and wherein each of the plurality of processors is associated with one of a plurality of local run queues (Column 24, lines 37-55: each node along the tree gets a queue, which includes the root nodes that are the processors), the method comprising the computer implemented steps of: performing, by an idle processor of the plurality of processors a first multiprocessor module of the plurality of multi-processor modules, a first attempt at a thread steal from a first one of the plurality of local run queues of one of the plurality of processors located on the first multi-processor module for reassignment of a thread to a second one of the plurality of local run queue associated with the idle processor (Column 11: lines 21-27: stealing within a module); responsive to failure of the first attempt, performing a second attempt at a thread steal from one of the plurality of chip run queues associated with a second multiprocessor module of the plurality of multi-processor modules and assigning the first thread to either one of the plurality of chip run queues or one of the plurality of local run queues (Column 10, lines 60-63; Column 11, lines 27-40: stealing amongst modules).

Final Office Action dated May 29, 2008 pp. 6-7.

Kimmel discloses utilizing a software abstraction of a non-uniform memory access system hardware representing a hierarchical tree structure to maintain the most efficient level of affinity and to maintain balanced processor and memory loads. The hierarchical tree structure includes leaf nodes representing the job processors, a root node representing at least one system resource shared by all the job processors, and a plurality of intermediate level nodes representing resources shared by different combinations of the job processors. Processors can belong to multiple run queues in the tree hierarchy, and threads can be dispatched at any level in the tree. Rebalance of nodes within the tree proceeds uninhibited all the way up the tree.

III.A. Claim 7

Claim 7 has been amended. Support for the claim amendments can be found on pages 23-27 of the specification. Claim 7, as amended is as follows:

> 7. A method of load balancing threads among processors in a multiple processor system having a plurality of multi-processor modules, wherein each of the plurality of multi-processor modules comprises a plurality of processors, wherein each of the plurality of multi-processor modules is associated with one of a plurality of chip run queues, and wherein each of the plurality of processors is associated with one of a plurality of local run queues, the method comprising the computer implemented steps of:
>
> performing, by an idle processor of the plurality of processors of a first multi-processor module of the plurality of multi-processor modules, a first attempt at a thread steal from a first one of the plurality of local run queues of one of the plurality of processors located on the first multi-processor module for reassignment of a thread to a second one of the plurality of local run queue associated with the idle processor;
>
> responsive to performing the first steal attempt, stealing the thread from the first one of the plurality of local run queues if the first one of the plurality of local run queues has a largest number of threads of all the local run queues of the multi-processor module,

if the first one of the plurality of local run queues contains more threads than an intra-module steal threshold of the first multi-processor module, and if the thread is an unbound thread;

responsive to failure of the first attempt, performing a second attempt at the thread steal from one of the plurality of chip run queues associated with a second multi-processor module of the plurality of multi-processor modules;

responsive to performing the second steal attempt, stealing the thread from the one of the plurality of chip run queues associated with the second multi-processor module of the plurality of multi-processor modules, if the one of the plurality of chip run queues has a largest number of threads of all of the plurality of chip run queues of the multi-processor system, and if the one of the plurality of chip run queues contains more threads than an inter-module steal threshold of the first multi-processor module; and

responsive to stealing the thread from either the first one of the plurality of local run queues or the one of the plurality of chip run queues associated with a second multi-processor module, assigning the thread to a chip run queue of the idle processor, or a local run queue of the idle processor.

Consistent with the remainder of the invention, amended claim 7 attempts to steal threads that will not result in a process switch or a context switch. Therefore, according to amended claim 7, a thread is stolen from a local run queue of a processor on the same multi-processor modules only "if the first one of the plurality of local run queues has the largest number of threads of all the local run queues of the multi-processor module, if the first one of the plurality of local run queues contains more threads than an intra-module steal threshold of the first multi-processor module, and if the thread is an unbound thread." Additionally, a thread is stolen from the chip queue of a different multi-processor module only "if the one of the plurality of chip run queues has the largest number of threads of all the chip run queues of the multi-processor system, and if the one of the plurality of chip run queues contains more threads than an inter-module steal threshold of the first multi-processor module." These features are completely absent from the disclosure of *Kimmel*.

Because *Kimmel* does not teach these claim features, *Kimmel* cannot properly anticipate amended claim 7 under 35 U.S.C. § 102. Withdrawal of the rejection is therefore respectfully requested.

III.B. Claim 17

Claim 17 has been amended to recite features similar to those found in claim 1. The remarks presented above with regard to claim 1 are equally applicable to claim 17.

Kimmel does not mention bound threads, nor does *Kimmel* offer any way to process a bound thread. Therefore, *Kimmel* does not disclose either of the claim 1 features of "identifying whether the first thread is a bound thread associated with a first processor of the plurality of processors," and "responsive to identifying that the first thread is a bound thread associated with a first processor, assigning the first thread to a first local run queue of the plurality of local run queues, wherein the first

local run queue is associated with the first processor." Because *Kimmel* does not teach these claim features, *Kimmel* cannot properly anticipate amended claim 1 under 35 U.S.C. § 102.

Kimmel does not disclose either of the claim 1 features of "attempting to identify an idle processor that has executed a second thread belonging to a first process, wherein the idle processor is identified from the plurality of processors of the first multi-processor module associated with the existing process," and "responsive to identifying the idle processor that has executed a second thread belonging to a first process, assigning the first thread to a second local run queue of a plurality of local run queues, wherein the second local run queue is associated with the idle processor." Because *Kimmel* does not teach these claim features, *Kimmel* cannot properly anticipate amended claim 1 under 35 U.S.C. § 102. Withdrawal of the rejection is therefore respectfully requested.

III.C. Claims 8-12 and 18-20

The rejections of claims 8-12 are predicated upon the assertions made with respect to independent claim 7. As shown above, the underlying assertions made by the Examiner regarding *Kimmel* are incorrect vis-à-vis independent claim 7 as amended. Therefore, *Kimmel* does not teach all of the features of claims 8-12 at least by virtue of their dependence on independent claim 7. Therefore, the rejection of claims 8-12 under 35 U.S.C. § 102 has been overcome. Withdrawal of the rejections is therefore respectfully requested.

The rejections of claims 18-20 are predicated upon the assertions made with respect to independent claim 17. As shown above, the underlying assertions made by the Examiner regarding *Kimmel* are incorrect vis-à-vis independent claim 17 as amended. Therefore, *Kimmel* does not teach all of the features of claims 18-20 at least by virtue of their dependence on independent claim 17. Therefore, the rejection of claims 18-20 under 35 U.S.C. § 102 has been overcome. Withdrawal of the rejections is therefore respectfully requested.

IV. <u>35 U.S.C. § 103, Obviousness</u>

The Final Office Action has rejected claim 4 under 35 U.S.C. § 103 as being unpatentable over *Alfieri*, U.S. Patent No. 5,745,778, (hereinafter "*Alfieri*"). This rejection is respectfully traversed.

Regarding this rejection, the Final Office Action states:

> As per claim 4, Alfieri teaches being able to identify threads having the same processor and multi-processor module (Column 6, lines 31-36; Column 9, lines 55-57; Column 10, lines 1-3). But Alfieri does not specifically teaches wherein the step of identifying the first multi-processor module further comprises: maintaining a record of processes having threads executed by a processor of the first multi-processor module during a predetermined preceding interval. However, it would

> have been obvious to one having ordinary skill in the art at the time of the applicant's invention to see that in order for Alfieri's invention to keep track which processor must be used to execute an application, there must be a record that records processes executed by a processor.

Final Office Action dated May 29, 2008 p.9.

The rejection of claim 4 is predicated upon the assertions made with respect to independent claim 1. As shown above, the underlying assertions made by the Examiner regarding *Alfieri* are incorrect vis-à-vis independent claim 1 as amended. Therefore, *Alfieri* does not teach all of the features of claims 4 at least by virtue of their dependence on independent claim 1. Therefore, the rejection of claims 4 under 35 U.S.C. § 102 has been overcome. Withdrawal of the rejection is therefore respectfully requested.

V. **Conclusion**

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: August 27, 2008

Respectfully submitted,

/Brandon G. Williams/

Brandon G. Williams
Reg. No. 48,844
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants